OMPUTER CORPORATION OF AMERICA
5 TECHNOLOGY SQUARE • CAMBRIDGE, MASS. 02139 • TEL. 617/491-3670

Semi-Annual Technical Report

September 1, 1971

6

ARPA Order Number	1731
Program Code Number	61101D
Name of Contractor	Computer Corporation of America
Effective Date of Contract	8 January 1971
Contract Expiration Date	7 January 1972
Contract Number	DAHC04-71-C-0011
Principal Investigator and Phone Number	Thomas Marill, 617-491-3670
Project Scientist and Phone Number	Kenneth Curewitz, 617-491-3670
Short Title of Work	Network Data Handling System
Amount of Contract	$229,955.00

Sponsored by

Advanced Research Projects Agency

ARPA Order No. 1731

Summary

The technical problem is concerned with providing data handling within a computer communications network. Elements of this program include a data language, a large capacity storage device and a special hardware system. The problem, then, for these program elements, is to design and develop a data language which allows network participants access to remote data; to use and investigate a large capacity storage device as primary storage for the system; and to investigate the use of a special hardware system (the datacomputer) as a data utility within the network for both remote and local users.

The method selected to solve the technical problem is first to establish user applications, techniques for using the laser memory system, and the specific hardware components of the system. These efforts are followed by hardware acquisition, component integration and design of the programming system which will support the applications on the hardware system. The next procedures include implementing the software and testing actual user applications against the datacomputer system. Finally, results of this testing will be evaluated to produce the information to satisfy the stated program objectives.

With regard to the results of this procedure, several major milestones have been achieved. Specifically, the major hardware components of the system have been identified and procurement procedures have been instituted for their timely delivery. A continuing dialogue has been established with prospective users and suppliers of the network data base to allow the specification of the basic access methods for the datacomputer. The system architecture for the datacomputer system (which is the basic structure of those programs which will accept the data language as presented by the users and will then act upon the

language to provide the requested actions) has been identified and is described in detail in the body of this report. Finally, specific procedures have been designed which allow the basic storage device of the datacomputer, that is, the laser mass memory system, to provide all features necessary for its efficient and effective use. This design effort represents a marked improvement over previously available techniques for storage and retrieval of information using this particular device.

The implication of current progress on the project is positive and far reaching. It appears certain that the hardware selected for the datacomputer will enable the very first such data utility within a computer network. The system architecture and specific techniques designed for the system guarantee the basic tools and procedures necessary to implement the datacomputer concept. Given the hardware components and the program structure for the datacomputer, it is then a matter of implementing the features of the design to begin investigation of a completely new technology in the use of computers and information systems.

While current objectives are very broadly specified, as indicated in the previous statement of the problem, there are certain logical extensions of those efforts which should be considered as candidates for further work. Where the current program is concerned with a single large capacity storage device and a single datacomputer servicing network participants, consideration should be given both to different types and multiples of large storage devices, as well as to multiple datacomputers within a single communications network. The latter proposal is an especially significant one since it would seem to allow for such features as graceful performance degradation, backup facilities and optimal use of any existing communications networks via proper routing and storage of information.

These are obvious extensions to the current work objectives based upon applying current procedures and thoughts to existing equipments and systems. It is expected that a completely new set of problems and solutions will become obvious once participants in the network begin using the datacomputer to provide on-line information with which to solve a new spectrum of computer applications.

Table of Contents

Preface

The present document discusses the software architecture of the datacomputer system. Four processors are jointly involved in the running of this software: A Hewlett-Packard HP2116B, two Word Processors, and a PDP-10. The first three of these are physically part of a Unicon 690 system which is incorporated within the datacomputer.

This document does not discuss the following:

- The data language through which external users interact with the datacomputer.
- The data management services offered by the datacomputer.
- Retrieval mechanisms and associated file structures.

These topics will be covered in subsequent working papers.

(N.B.: The present document is subject to revision without notice.)

Chapter 1
Overview of the Datacomputer

1.1 Outline of the Datacomputer

The datacomputer is a black box having two principal hardware ports, Port 0 and Port 1. Port 0 connects to an IMP; Port 1 connects to the Illiac IV system (see Fig. 1.1). Inside the black box are various hardware components of which the principal three are: PDP-10, UNICON 690, and disks.

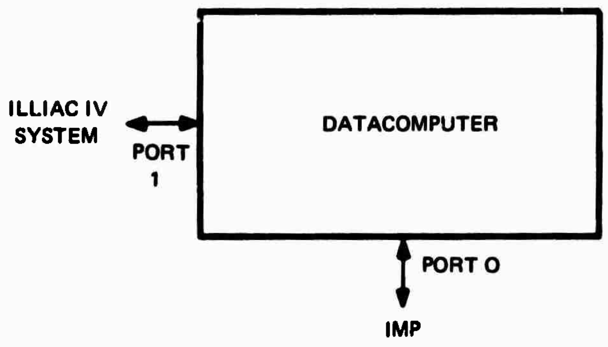

Figure 1.1 - External View of Datacomputer

Two kinds of information travel through the ports: command information and data. The command information primarily consists of requests for data management service.

1.2 Datacomputer Software Modules

The datacomputer software consists of four modules: input-output manager, request handler, supervisor, and storage manager services (see Fig. 1.2).

1. Input-Output Manager. The input-output manager handles communication between the datacomputer and "sequential devices": IMP, Illiac IV, magnetic tape, reader/punch, and printer.

On output, the job of the input-output manager is to accept logical records from other modules and to convert them into proper units for transmission to the appropriate sequential device. On input, the job is the reverse.

2. Request Handler. The request handler consists of three main modules: command executor, compiler, and interpreter. The command executor executes various commands that need not be compiled. The compiler compiles a source request, expressed in datalanguage, into a compiled request, expressed in terms of datacomputer primitives. The interpreter interprets the compiled request. Each request is considered an independent process, which interrupts when waiting for the execution of commands to the I/O manager or storage manager. The scheduling of these processes is performed by the supervisor.

3. Supervisor. The supervisor performs a number of functions, the primary one being the scheduling of request processes and other independently running processes in the I/O manager and the storage manager. The supervisor is also responsible for system and process initialization.

4. Storage Manager. The storage manager allocates storage space on and causes data transfers between core buffers, disks, and strips.

The storage manager accepts commands from the other three modules.

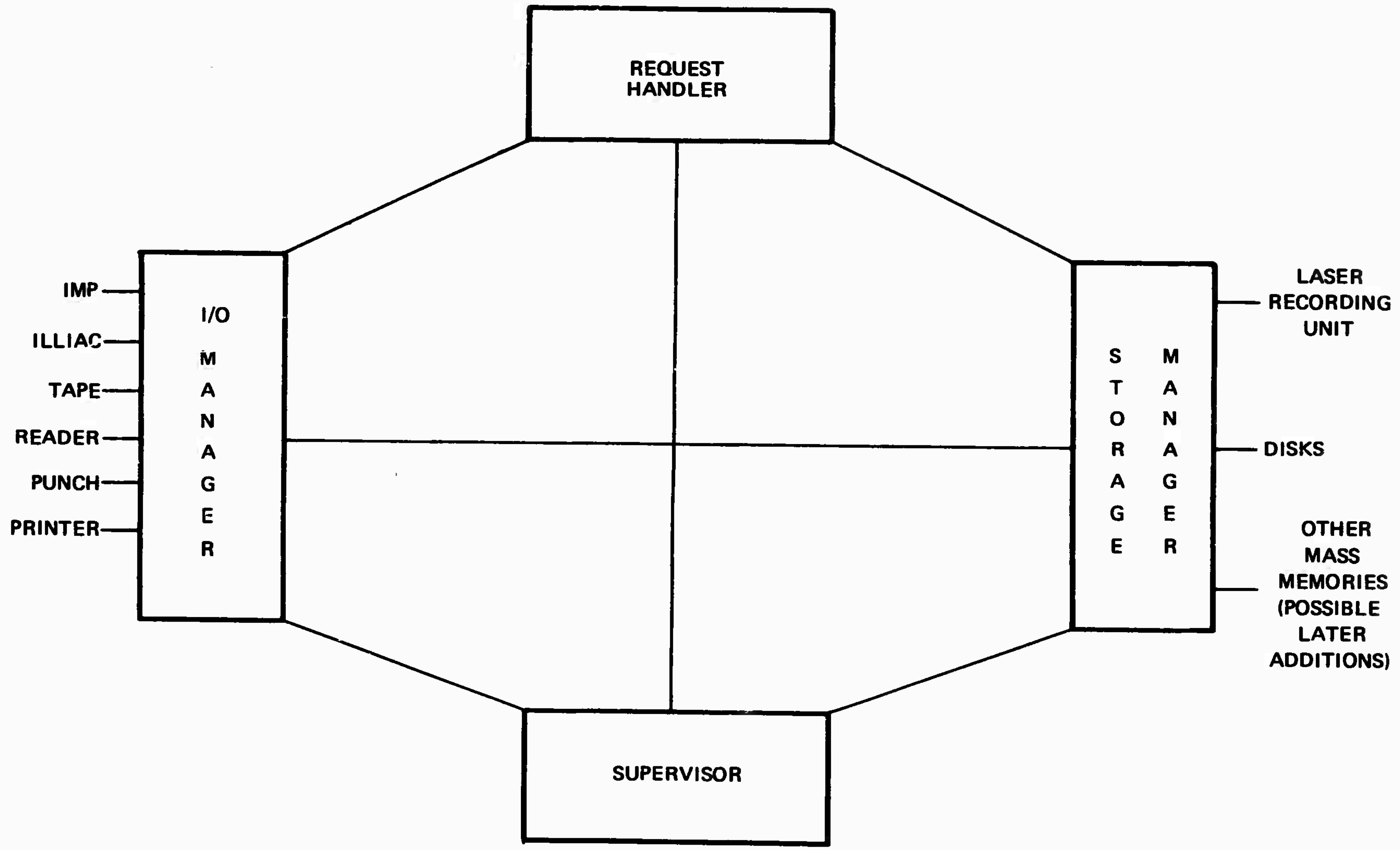

Figure 1.2 - Architecture Overview

The job of the storage manager is (a) upon request, to make data available in a core buffer as soon as possible and to so inform the requesting process; (b) to write data from core buffers to disk and strips in response to commands; and (c) to allocate space on various storage devices.

Other modules may tell the storage manager of the future data demands they anticipate. The storage manager monitors activity in other parts of the system and uses this information in scheduling strip mounts and data transfers. The Hewlett-Packard software and the word processor software in the UNICON 690 are considered part of the storage manager module.

1.3 File Structures

At different stages of processing, a file in the datacomputer is described in terms of different file structures. These file structures are described below.

User-Defined File Structures.

The user-defined file structure is entirely independent from datacomputer file structure; the user designs his files to meet his particular needs. In requests to the datacomputer, the user defines and refers to his files in terms of his own, user-defined file structure. The datacomputer's request handler requires that, whenever a file is created, the user specify a file structure definition for that file. A file structure definition is a definition of the logical format of records in the file (such as varieties and properties of fields that may occur in a record).

The user also specifies the physical and logical properties of the data which is transmitted between the user and the datacomputer. In addition to the logical format of the transmission, the user specifies the physical format of records transmitted (such as the number of bits per field and the physical denotation of record boundaries).

Datacomputer File Structures.
Users' files are stored in the datacomputer on addressable pages. To store or retrieve a user's file or part of a file in the datacomputer, the addresses of the pages corresponding to the file or part of a file are determined in two steps:

1. The user's reference, in a request, to a file or part of a file is converted by the request handler to logical page addresses (LPAs) of the pages to which the file or part has been assigned. An LPA is the concatenation of five numbers:

FILE:SECTION:SUBSECTION:AREA:PAGE

An LPA specifies a page address in the datacomputers' logical page space. The request handler uses the file description supplied by the user when the file was created and auxiliary tables (such as inverted files and indices) to make the conversion from a user file reference to an LPA. Pages of the auxiliary tables are also expressed by LPAs.

2. The storage manager maps LPAs into physical page addresses (PPAs), which are addresses of physical locations in the datacomputers' storage devices. The set of the locations in the storage devices is the datacomputer's physical page space.

Chapter 2
Request Handler

2.1 Overview

A user logs into the datacomputer through a special logical network port to the control module in the supervisor. The control module calls the process initiator, which initiates a user process in the datacomputer for the user.

The process initiator (1) creates a process state table entry for the user process and (2) creates a port monitor for receiving commands from the user.

Once initiated, a user process is an independent process, whose execution is scheduled by the supervisor. The supervisor selects which user process is to be run.

The command interface module of the request handler receives and dispatches user commands for the user process selected by the supervisor. Commands are received through the port created during process initiation and are dispatched by the command interface module to the command executor until a BEGIN command is encountered. Commands following the BEGIN are sent to the compiler until a matching END command is encountered; then the commands between the BEGIN and the END are compiled, and the compiler calls the interpreter, which interprets the compiled commands.

Execution of commands and interpretation of compiled commands involve computation and calls to other datacomputer modules (e.g., the storage and I/O managers). Execution of a user process may be stopped temporarily when (1) the user process is waiting for I/O from a port monitor; or (2) the user process is waiting for completion of a storage manager command.

Further design of the request handler is dependent on specification of the datacomputer's access methods and of the language in which requests will be expressed; both will be described in subsequent working papers.

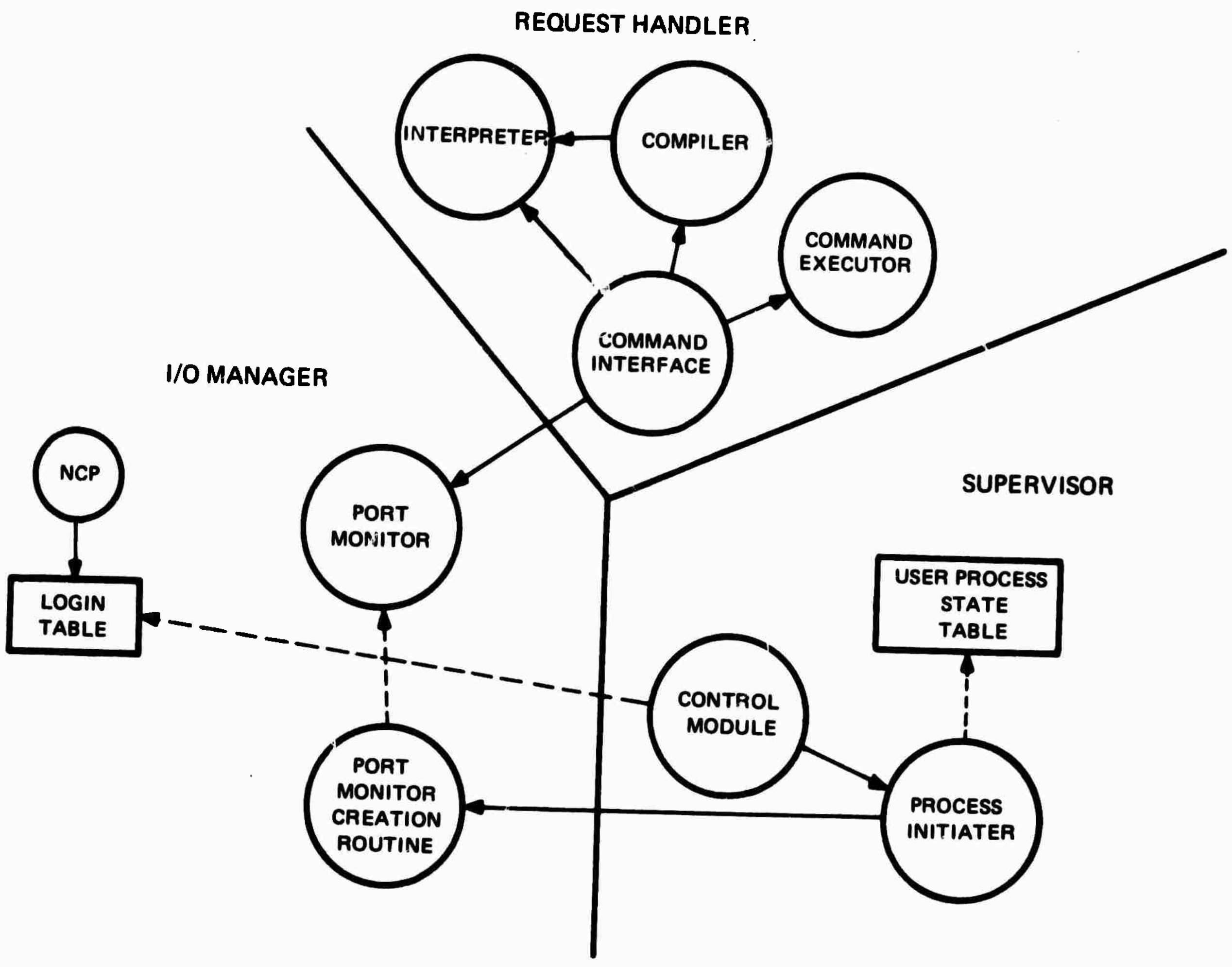

Figure 2.1 - Process Creation

Chapter 3
The Storage Manager

3.1 Storage Manager Interface

Functions Performed by the Storage Manager.
The functions of the storage manager fall into four broad classes:

a. Handling Data File Requests--The storage manager accepts commands to read and write pages from data files expressed in terms of an internal logical file structure.

b. Handling Scratch Pages--The storage manager allocates pages in the basic page space.

c. Managing Core Buffers--The storage manager controls the swapping of basic pages and data pages to and from core buffers.

d. Allocating Data File Areas--Storage for internal logical files is allocated on the datacomputer's mass memory devices.

The storage manager commands will be presented following a discussion of some storage manager interface concepts.

The Internal Logical File Structure (Logical Page Space).
Data requests to the storage manager are expressed in terms of an internal logical page space. Each data file stored on the datacomputer is expressed as a file of this structure. The request handler is responsible for converting to the internal logical page structure from the external logical file structure in which the user refers to a file. Data pages are fixed length pages of 1K words.

Logical page addresses (LPAs) in a data file are expressed as a concatenation of five numbers:

FILE:SECTION:SUBSECTION:AREA:PAGE.

Some of the storage manager commands require that an entire data page group (i.e., file, section, subsection, or area) be specified. To allow for this, numbering begins at 1 instead of 0. 0's in lower level positions indicate the entire data page group. Thus, 327:3:2:0:0 indicates a data page group consisting of all pages in file 327, section 3, subsection 2.

The storage manager translates the internal logical page space into a physical page space, which remains invisible to other programs in the datacomputer.

Basic Page Space.
The storage manager maintains a set of 1K pages in an abstract page space called the basic page space. Each page has a unique identifier called the page id (PID). PID = 0 is not used. Occasionally these pages reside in core buffers where they may be used by datacomputer programs.

Page Address Words (PAW).
Several of the storage manager commands involve the use of PAWs. They are blocks of storage and are used to transmit arguments to the storage manager and to return results from completed storage manager commands.

A PAW consists of two fields. The first is a page id of a page in the basic page space. The second is the address of the buffer in which that page resides if it is in core.

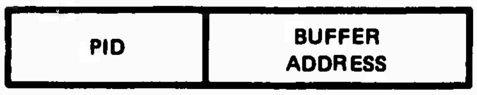

Figure 3.1 - Page Address Word

The READ-CHOICE command discussed below requires a modified PAW, called an extended PAW or XPAW. In addition to the PID and buffer address field, the XPAW contains a field for a logical page address.

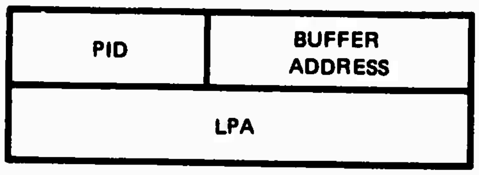

Figure 3.2 - Extended Page Address Word

The Storage Manager Commands.
In the following, the commands which may be issued to the storage manager are each followed by a description of the action taken in implementing the command.

ASSIGN<n, pointer to PAW>
n contiguous pages in the basic page space are assigned to the user. The PAW is zeroed. The page id of the first page in the block is placed in the PID field of the PAW.

If it is not possible to assign n contiguous pages, the storage manager returns, leaving zero in the PID field.

RESTORE <pointer to PAW, new page bit>
The page with the PID specified in the PAW is brought into a core buffer. The storage manager returns immediately to the calling program. This command requires a corresponding WAIT command. Upon return from the corresponding WAIT command, the storage manager will have placed the address of the core buffer in which the page resides in the buffer address field of the PAW.

If the new page bit is set, then a new version of the page is placed in the core buffer. This page is not necessarily zeroed.

SHARE <pointer to PAW>
This command indicates to the storage manager that the page indicated in the PAW is shared by two separate processes*. STANDBY and RESTORE commands for the same page may then be issued independently by each process. Two RELEASE commands will be required before the page is actually freed.

NOTEMOD <pointer to PAW>
This command is executed for in-core pages to inform the storage manager that the page will be modified. This command must precede the first modification of a page following a RESTORE command for that page. The PAW is examined to determine the page id and buffer location.

STANDBY <pointer to PAW>
This command releases the core buffer specified in the buffer address field of the PAW. The PID field of the PAW should specify the page which actually resides in the buffer. The buffer address field in the PAW will be zeroed.

Note that a page which has been "stoodby" can be brought back into core with a RESTORE command.

* Such usage may exist, for example, between a process being executed by the request handler and an associated port monitor process in the I/O manager.

RELEASE <pointer to PAW>
This command releases both a page in the basic page space and the buffer in which it resides. The page indicated in the PID field of the PAW will become available for assignment to other users. If the buffer address field is non-zero, then the buffer will also be freed as in a STANDBY. The PAW will be zeroed.

READ <pointer to PAW, LPA, FR bit>
The contents of the data page specified by the logical page address are read into a page in the basic page space. The basic page will reside in a core buffer. This command requires a corresponding WAIT command. Upon return from the WAIT the data page will reside in a core buffer whose address will have been placed in the buffer address field of the PAW. A page id in the basic page space will be assigned to the page. If the PID field in the PAW is non-zero, then that page id will be used. Otherwise a new page will be assigned and its page id placed in the PID field of the PAW. By using this page id, the page may be STANDBYd, RESTOREd, RELEASEd, and STOREd. Use of the FR bit is explained under the command FUTURE READ.

READ-CHOICE <pointer to XPAW, LPA1,...,LPAn, FR bit>
This command has the same effect as a READ. The storage manager is given the choice, however, of which data page to read. The data page which is chosen will be indicated in the LPA field of the XPAW. Use of this command will allow the storage manager to optimize the order of retrievals in the request handler. This command requires a corresponding WAIT as for the READ command. Use of the FR bit is explained under the command FUTURE READ.

FUTURE READ <LPA>
This command indicates that a READ or READ-CHOICE command will follow for the indicated data page. In this case the corresponding READ or READ-CHOICE must have the FR bit set. If the FR bit

is set in a READ-CHOICE command, then all of the LPAs listed in that command must have been FUTURE READ. If the FR bit is not set, then none of the LPAs listed may have been future read. Use of this command will allow the storage manager to optimize retrieval of data pages.

STORE <pointer to PAW, LPA>
The contents of the page indicated by the PID field will be stored in the data page designated by its LPA. The page must reside in a buffer. Both the page and buffer will be released and the PAW will be zeroed.

FUTURE STORE <LPA>
This command serves to notify the storage manager that a STORE command for the indicated data page is forthcoming.

WAIT
The WAIT-requiring commands are RESTORE, READ, READ-CHOICE, and XPDA. When a WAIT command is received, the storage manager will not return immediately but instead will wait until all WAIT-requiring commands that have been issued since the last WAIT have been completed. During this time the storage manager will call the supervisor so that it may run other processes.

ADD <data page group, partial internal logical file description>
The partial internal logical file description specifies the attributes of a new page group for the file. This page group will be allocated subordinate to the specified data page group. In the case that data page group is given as 0:0:0:0:0 a new file will be allocated. Other processes may run while space is being allocated.

DELETE <data page group>
The data page group will be deleted. File, section, subsection, and area numbers so deleted may be reused. Other processes may run while the data page group is being deleted.

XPDA <n>

The mnemonic stands for extend process data area. The process data area is that set of pages on which the process-private data for a process is kept. The process pages provide a data area for the datacomputer's pure procedures. Whenever a process is being executed, its process pages are kept in contiguous core locations. The XPDA command directs the storage manager to expand (or contract) the size of this area by n pages (where n is a signed integer). This command requires a corresponding WAIT.

3.2 Storage Manager Architecture

Introduction.

The storage manager is capable of storing data on pages, on both disk and laser memory. Each page will have a physical page address (PPA) which directly addresses its location on a device. The interface to the storage manager references data pages by their LPAs. These are converted to PPAs by the storage manager. The abbreviation dPPA will be used to indicate specifically a disk page address. Disk space will be used for both data and temporary pages. Data pages with corresponding LPAs reside on the disk data pages. It will be possible to distinguish between the use of a disk page as a data or non-data page by its dPPA.

When a core buffer is assigned, it will also be assigned a disk page to which it can be swapped when not in core. The interface refers to the assigned disk page by a PID; the PID can be trivially mapped into the dPPA of the disk page. Both disk data and non-data pages can be referenced in PIDs.

Figure 3.3 is a general overview of the storage manager architecture. In this and following diagrams, circles represent modules and rectangles represent tables. Solid arrows indicate directed flow between modules; dotted arrows indicate reading and modifying of tables.

A discussion of the functions performed by each module follows the descriptions of the storage manager tables. A glossary of acronyms is included as an appendix.

Storage Manager Tables.

The Buffer Usage Table (BUT).

The BUT has one entry per buffer. Each entry has the format:

<dPPA> <CNT><being-read bit> <change bit> <UBB>

The dPPA field specifies the address of the basic page which resides in the buffer. If it is zero, then the buffer is empty (but not necessarily all zero). The dPPA may specify either a disk data page or a disk non-data page.

A buffer may be shared by a number of processes; the CNT field indicates how many. In the case of a non-data page, this CNT should never be greater than two. In the case of a disk data page, however, any number of processes may be using the buffer. When the page is STANDBYd or RELEASEd, the CNT is decremented. When the page is RESTOREd the CNT is incremented. If the CNT is 0, then the buffer is not being used. Note that, if dPPA field is non-zero while the CNT is 0, the buffer contains a copy of the indicated page.

The being-read bit is set when a page has been assigned to the buffer but the buffer does not contain an accurate copy of the page because it is still in the process of being read.

The change bit indicates whether or not the buffer contents are the same as the contents of the disk page specified in the dPPA field. If the bit is 0, the contents are the same. If CNT = 0 and the change bit is not set, then the buffer can be assigned to another page.

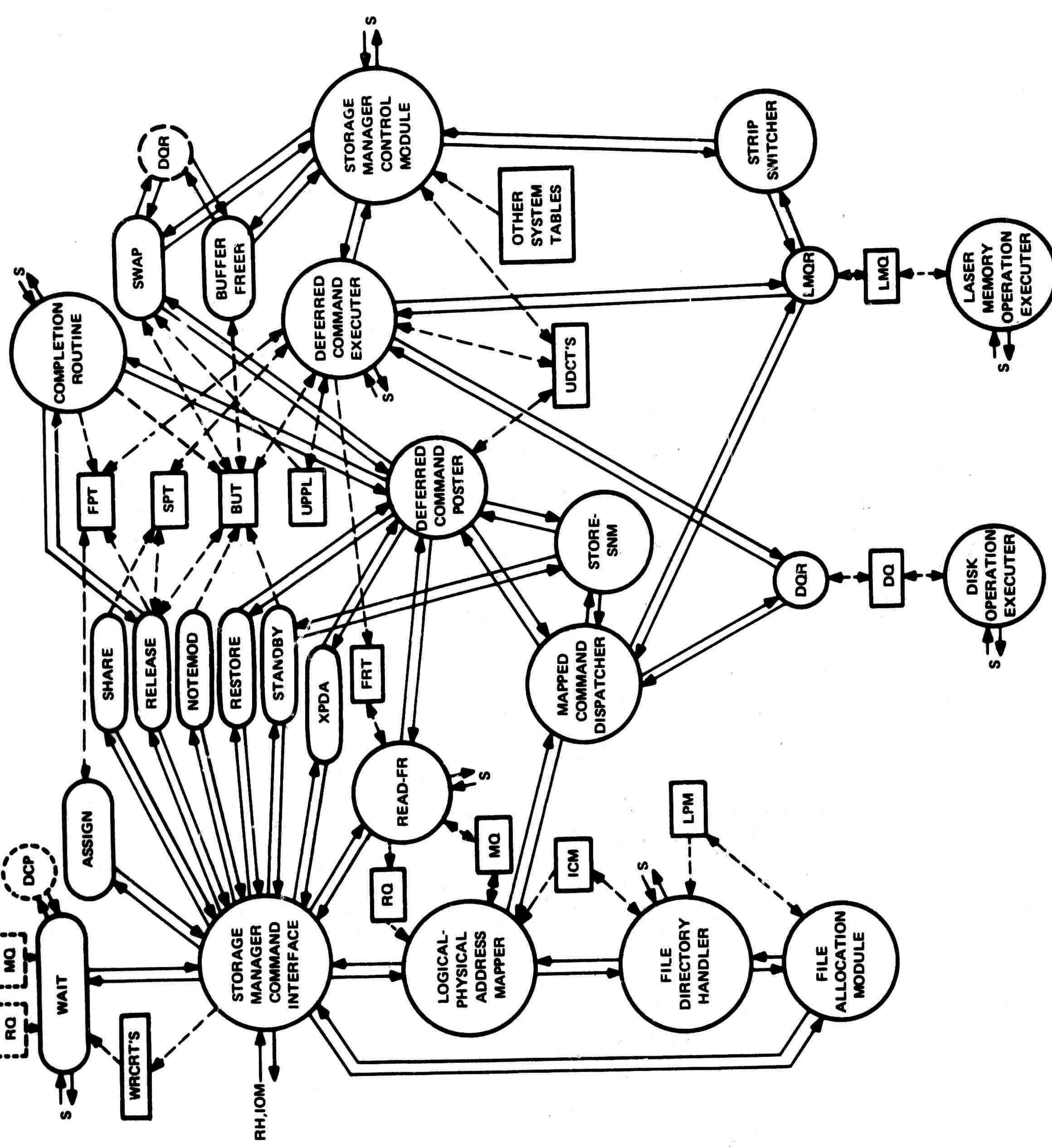

Figure 3.3 - Storage Manager Architecture

The UBB field is a usage bit block. Each bit represents a datacomputer process. If a particular process' bit is on, then that process is using the buffer. The number of bits which are set in the UBB should be the value of the CNT field.

The Free Page Table (FPT).
This table is a bit pattern containing one bit for each non-data disk page. If the bit corresponding to a disk page is set, then that page is in use and cannot be assigned.

The Shared Page Table (SPT).
This is also a bit pattern with one bit per non-data disk page. If the bit corresponding to a page is set, then that page is in shared mode. This table is used in conjunction with the FPT to insure that two RELEASEs are given before the page is actually freed. Note that neither the FPT nor SPT need cover disk data pages since these are never actually freed except through the DELETE command.

The User's Process Page List (UPPL).
For each process, an ordered list is kept of the dPPAs on which its process pages reside. This list is the UPPL.

The Wait-Requiring Command Results Table (WRCRT).
Each process entry will have a number of WRCRT entries of the format

    <Addr> <XPAW Image>

Whenever a wait-requiring command is received by the storage manager, a WRCRT entry is allocated. The address of the WRCRT entry is appended to the command. The address of the user's PAW or XPAW is placed in the Addr field of the WRCRT. When results are produced in the execution of the wait requiring command, they are placed in the XPAW Image field. Before returning from the WAIT command, the results are transferred from the XPAW Image field into the user's PAW or XPAW. This mechanism is necessary since the user's PAW or XPAW may be swapped out of core while his commands are being executed.

The Future Read Table (FRT).
This is a hash table which is kept on disk. Each entry is of the form

    <LPA> <dPPA>

The entries are hashed on the LPA field. An FRT entry is made when a page is placed on the disk as the result of a FUTURE READ command. When a READ command for a FUTURE READ page is received, the FRT is examined to see where the page is stored on disk. FRT entries are removed upon receipt of the READ command.

The Logical-Physical Map (LPM).
This is the table which contains the mapping from internal logical page space to physical page space. Its format is as yet undetermined and will depend upon the nature of file structures and space allocation algorithms. It will reside on disk pages and will enable the translation of any LPA into the proper PPA.

The In-Core Map (ICM).
This is a mini-version of the LPM, which is maintained in core. It should operate analogously to a cache memory. That is, recently accessed portions of file maps will be kept in the ICM so that disk transfers are not needed to bring the LPM into core for each page mapping.

The Read Queue (RQ).
Since execution of READs which had corresponding FUTURE READs will require disk accesses to obtain FRT pages, the READ commands are held in the RQ until the needed FRT page is available in core.

The Mapper Queue (MQ).
This is used in a manner similar to the RQ to hold commands until the needed LFM page can be brought into core.

The User's Deferred Command Table (UDCT).
Each user has a UDCT. In addition, those system processes which issue commands to the storage manager have UDCTs. Wait-requiring commands must always be deferred. Also STOREs to an unmounted strip must be deferred. FUTURE READs and FUTURE STOREs are placed on UDCTs. Commands on a UDCT are removed when they have been completed. Associated with each UDCT are three bits. The first two are the "wait issued" bit and the "wait" bit. The wait issued bit indicates that a WAIT command has been received from the user, but that there are still outstanding wait-requiring commands in the MQ or RQ. The wait bit indicates that the WAIT has been received and that only the UDCT contains wait-requiring commands. When all wait-requiring commands in the UDCT have been completed, the user process can be restarted. The third bit is the "system" bit. This bit is set for most system process UDCTs. If the system bit is set, then whenever any wait-requiring command in the UDCT is completed, the process can be restarted with an indication of which command was completed.

The Disk Queue (DQ).
This is a list of reads and stores to the disk. These are the UDCT commands to which buffers and disk pages have been assigned. The user id is also attached to the command so that, when the command has been completed, the appropriate UDCT can be modified.

The Laser Memory Queue (LMQ).
This queue is analogous to the DQ.

Storage Manager Modules.

The Storage Manager Command Interface (SMCI).

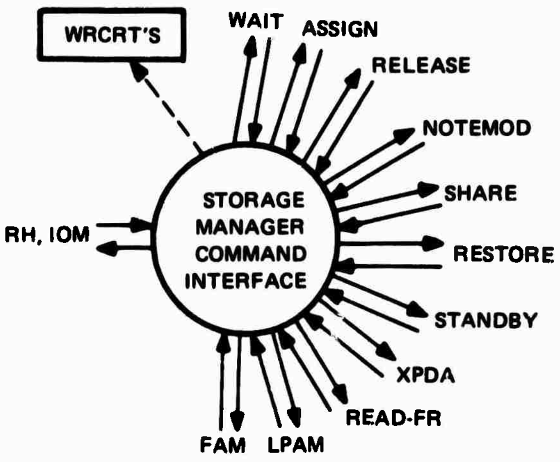

Figure 3.4 - Storage Manager Command Interface

This module accepts commands from the request handler and I/O manager and dispatches them to the appropriate module. For wait-requiring commands, the SMCI also establishes a WRCRT entry and appends its address to the command. Note that the READ-CHOICE is handled as if it were several separate READs. Only one of the READs will actually be performed.

The WAIT Module.

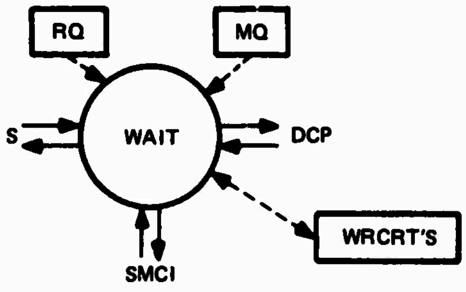

Figure 3.5 - WAIT Module

WAITs are dispatched to the WAIT module by the SMCI. If the RQ, MQ and UDCT are free of the user's commands, then restart conditions for the user are established. If only the RQ and MQ are free of the user's commands, the wait bit is set in the user's UDCT. Otherwise, the wait issued bit is set.

The supervisor is then called to switch processes. When the user process is returned to, it is restarted at the WAIT module. Before returning to the SMCI, the information in the WRCRTs is transferred to the user's PAWs.

The ASSIGN Module

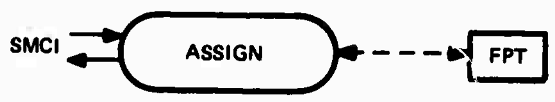

Figure 3.6 - ASSIGN Module

ASSIGN commands are sent through the SM command interface to the ASSIGN module. The ASSIGN module finds n contiguous free disk pages by consulting the FPT. It sets the corresponding bits in the FPT to 1's. The module then places the PPA of the first page in the block into the PID field of the user's PAW.

If it is not possible to assign n contiguous pages, the module sets the PID field in the user's PAW to zero.

The ASSIGN module returns to the SM command interface.

The RELEASE Module.

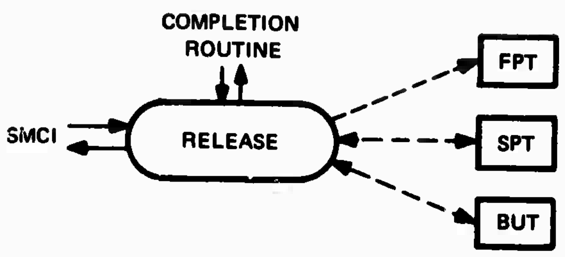

Figure 3.7 - RELEASE Module

RELEASE commands are sent through the SM command interface to the RELEASE module. The BUT entry corresponding to the buffer address in the user's PAW is examined, the CNT field is decremented and the process' UBB bit is cleared. If the dPPA field indicates a non-data page, then the SPT bit for that page is examined. If it is 1, then it is set to 0. If it is 0, then the corresponding bit in the FPT is set to 0 and the BUT entry is cleared. If the dPPA field indicates a data page and the change bit is off, then the BUT entry is not modified. (This allows other users to access the page from core if it still is available at the time their READ commands are executed.) If the dPPA indicates a data page and the change bit is on (note that the CNT must be zero), then the BUT entry is cleared to prevent other users from accessing a changed page.

The RELEASE module returns to the SM command interface.

The NOTEMOD Module.

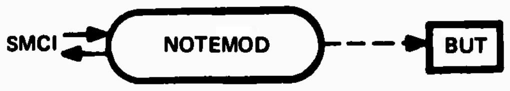

Figure 3.8 - NOTEMOD Module

NOTEMOD commands are passed to this module by the storage manager command interface. The module sets the change bit in the BUT entry corresponding to the buffer address field of the user's PAW and returns.

The SHARE Module.

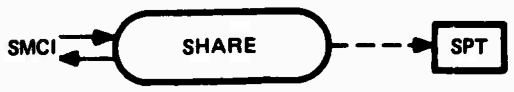

Figure 3.9 - SHARE Module

SHARE commands are passed by the SMCI to this module. The SPT bit corresponding to the PID field of the user's PAW is set.

The RESTORE Module.

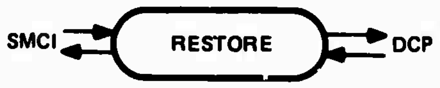

Figure 3.10 - RESTORE Module

RESTORE commands are sent to the RESTORE module, by which they are passed on to the deferred command poster, where they are placed in the user's UDCT. Upon return from the deferred command poster, the RESTORE module returns to the SM command interface.

The STANDBY Module.

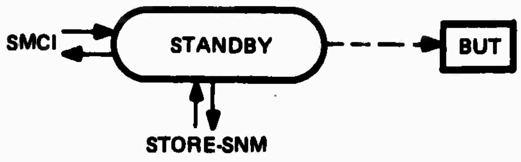

Figure 3.1˙ - STANDBY Module

STANDBY commands are sent through the SM command interface to the STANDBY module. The BUT entry corresponding to the buffer address in the user's PAW is found, the CNT field is decremented, and the process' UBB bit is cleared.

The STANDBY module returns to the SM command interface.

The XPDA Module.

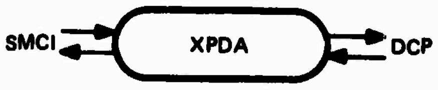

Figure 3.12 - XPDA Module

XPDA commands from the SMCI are passed on to the DCP to be placed in the UDCT.

The READ-FR Module.

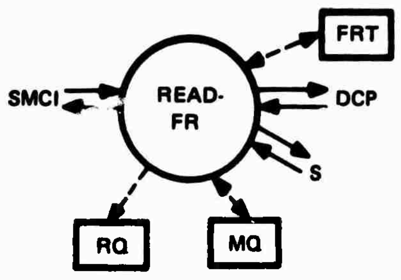

Figure 3.13 - READ-FR Module

READ commands with the FR bit set are dispatched to the READ-FR module by the storage manager command interface. First the module attempts to locate the unexecuted corresponding FUTURE READ command. It first checks the MQ. If it finds the FUTURE READ there, it is replaced by the READ command. If the FUTURE READ is not in the MQ, the user's UDCT is examined. If it is found there, the FUTURE READ is replaced by the READ command (note that address mapping will have already been performed*).

If the FUTURE READ is not found in the MQ or UDCT, the FUTURE READ must have been executed and a FRT entry established. A RESTORE request for the proper FRT page is placed on the READ-FRs UDCT. The system bit is set for this UDCT, so the READ-FR module is restarted whenever any of its requests are satisfied. While waiting for FRT pages to be read in, a READ command is held on the RQ.

---

* This is a tricky operation and requires careful planning, for example, to insure successful results if the FUTURE READ has been partially executed.

The Logical-Physical Address Mapper Module (LPAM).

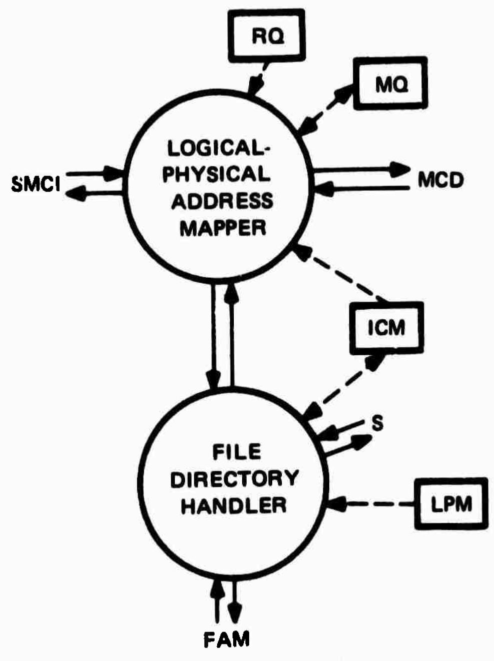

Figure 3.14 - LPAM Module

Commands that require LPA to PPA mapping are sent to the LPAM by the SM command interface. These include STORE, FUTURE STORE, FUTURE READ and those READ commands with the FR bit set to zero.

Core space is reserved to hold selected portions of the LPM in core. The entire LPM is stored on disk pages. If the LPA of a command is not included in the in-core part of the LPM, the LPA <u>cannot</u> immediately be mapped into a PPA; the command is placed on the mapper queue; the LPAM calls the file directory handler, which places a request for the LPM page which includes the LPA; and, after the file directory request is placed, the LPAM either returns to the SM command interface or proceeds to map the next command on the MQ.

If the LPA of a command is included in the in-core part of the

LPM and therefore can immediately be mapped into a PPA, the LPAM adds the PPA to the command and sends the mapped command to the mapped command dispatcher.

If the wait issued flag is set for a user's process and the LPAM has cleared the MQ of the user process commands, the LPAM checks the read queue to see if it is also clear of commands for the user process. If so, the wait issued flag is cleared and the wait flag for the process is set.

When the file directory handler has added the needed information to the in-core map, it will call the LPAM. The LPAM at this time maps commands from the MQ and sends the mapped commands to the mapped command dispatcher. Upon return from the mapped command dispatcher, the LPAM returns to the file directory handler.

The File Allocation Module (FAM).

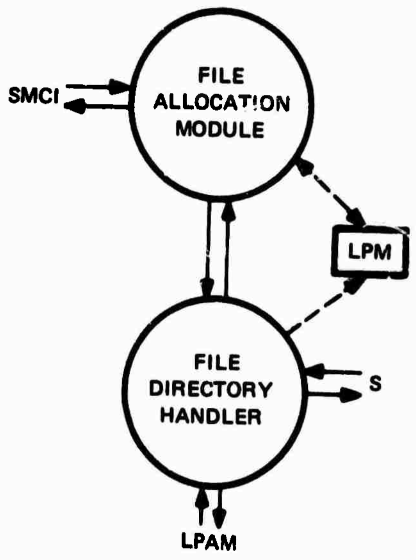

Figure 3.15 - FAM

ADD and DELETE commands are processed by this module. The FDH is called to bring needed pages of the LPM into core where they are updated by the FAM. File allocation strategies and the structure of the LPM will be the subjects of future working papers.

The Mapped Command Dispatcher Module (MCD).

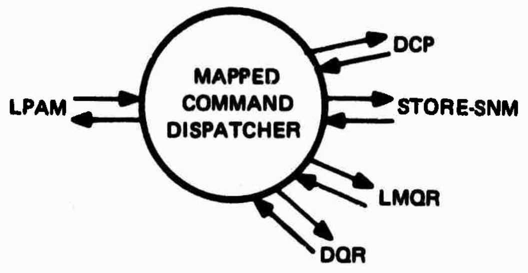

Figure 3.16 - MCD Module

The mapped command dispatcher module dispatches a mapped command to an appropriate queue or the DCP, which will place the command in the UDCT.

STORE Commands.
A STORE-to-disk command is added to the disk operations queue (DQ). A STORE-to-LM-for-strip-mounted command is added to the laser memory operations queue (LMQ). A STORE-to-LM-for-strip-not-mounted command is sent to the STORE-SNM module.

READ, FUTURE READ and FUTURE STORE Commands.
These commands are sent to the DCP to be placed in the UDCT.

After placing a command on the LMQ or DQ or after return from the DCP, the mapper command dispatcher returns to the LPAM module.

The STORE-SNM Module

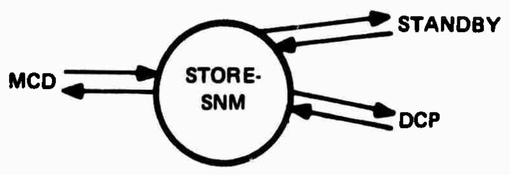

Figure 3.17 - STORE-SNM Module

The STORE-SNM receives from the mapped command dispatcher a command to STORE a page onto a laser memory strip which is not mounted. The STORE-SNM module calls the STANDBY module to put the page on STANDBY. On return from the STANDBY module, the STORE-SNM module replaces the STORE in the command operation field with RESTORE-and-STORE. The modified command is then sent to the DCP to be placed in the UDCT.

The Deferred Command Poster (DCP).

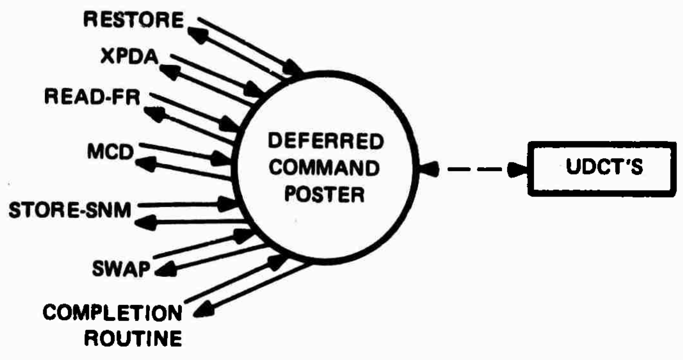

Figure 3.18 - DCP Module

The DCP is the module through which most other modules interrogate and modify the UDCTs, thus isolating the physical structure of the UDCT from these modules. The DCP can then cooperate with the SMCM in the ordering of commands on the UDCTs.

The Disk and Laser Memory Queuer Modules (DQR and LMQR).

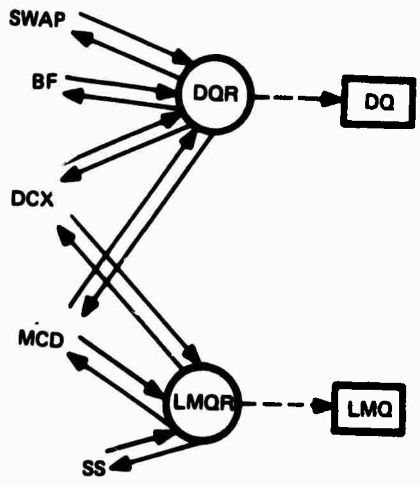

Figure 3.19 - DQR and LMQR

These modules place commands on the DQ and LMQ respectively. They can be designed to order the commands on the queues in an optimal sequence for the particular device.

The Deferred Command Executor (DCX).

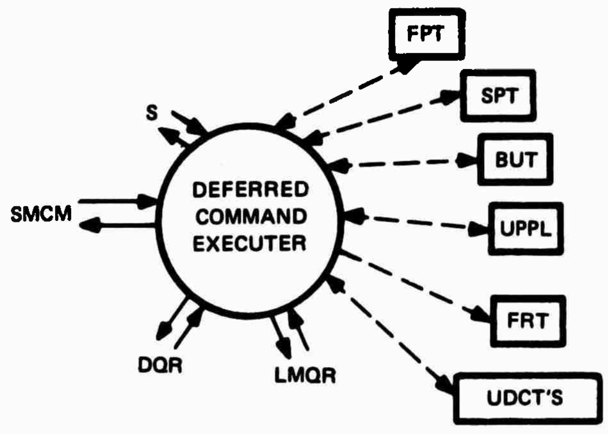

Figure 3.20 - DCX Module

This module is called by the SM control module to execute selected commands in some particular user's UDCT. The commands and user are designated by the SM control module, which has insured that disk pages and buffers are available for allocation to the commands. The functions performed for various commands are discussed below. Whenever a command is executed, a "being executed" flag is set for that command. This flag is cleared by the completion routine when the command has been completed.

RESTORE--The BUT is examined to see if the dPPA already resides in a core buffer. If so, the CNT field is incremented, the process' UBB bit is set, and the completion routine called to remove the command from the UDCT and to fill in the WRCRT entry.

If the dPPA does not reside in a core buffer, then a free buffer is chosen and the dPPA entry is filled, the CNT set to 1, the process' UBB is set, and the being read bit set. If the new page bit in the RESTORE command is set, then the change bit in the BUT is also set, and the completion routine called as above. If the new page bit is not set, then the RESTORE is transferred to the DQ so that the disk page may be read.

LM-READ--These must be laser memory reads for a currently mounted strip. If a PID had not been specified in the command, then a disk page is assigned and its FPT bit set. A buffer is selected, the dPPA field set, the CNT is set to 1, the process' UBB bit is set, and both the being read and the change bits are set. The command is then placed in the LMQ.

D-READ--If the PID had not been specified, then the dPPA of the data page is chosen as the basic page. The D-READ is then processed as if it were a RESTORE command.

If the PID had been specified, then that dPPA is placed in the BUTs dPPA field. The CNT field is set to 1, the process' UBB bit is set, and the being read bit is set. The D-READ is placed on the DQ to effect reading of the data page into core.

STORE-SNM--This is a laser memory store to a strip which had not been mounted. When the DCX is called to execute such a command, the strip-not-mounted condition should no longer be true. This command is executed in two parts. First the RESTORE cperation is performed on the page. When the page is in core, the completion routine will call the DCX to perform the second part. This involves placing a STORE command on the LMQ.

FUTURE-READ--This applies only to laser memory pages on the currently mounted strip. Like STORE-SNMs they also are processed in two parts. First a LM-READ sequence is executed. When the completion routine calls the DCX indicating that the LM-READ has been completed, the DCX establishes a FRT entry and then STANDBYs the page.

SWAPIN--This command is placed in the UDCT after a user's process pages have been swapped out. When executed, the process pages are brought back into core. The UPPL indicates the number and order of process pages to be swapped into contiguous core buffers. The SWAPIN module determines from the BUT whether enough free, contiguous buffers are available to accommodate the process pages. If there are enough free, contiguous buffers available, the process pages are read into the core buffers in the order specified in the UPPL. If there are not enough free, contiguous buffers available, then scratch pages in some buffers must be moved to free buffers so that a contiguous block of core buffers may be created. The DCX reorganizes the buffers and BUT when necessary.

XPDA--If process pages are to be freed (i.e., negative n) these pages and buffers are freed as in a RELEASE command.

If new process pages must be added to the contiguous buffer area, it may be necessary to reorganize the buffers as in the SWAPIN command. This operation will normally involve relocating buffer references in the process pages of those processes using a relocated page. Otherwise this process is similar to ASSIGN execution.

The Buffer Freer Module (BF).

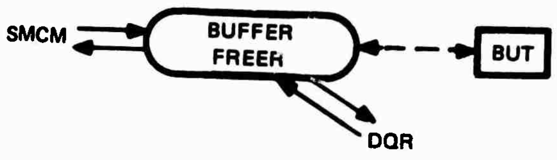

Figure 3.21 - BF Module

This module is called by the SMCM to initiate a disk write for a page in a buffer which has standby status (i.e., CNT = 0, change bit = 0). The BUT entry should have the change bit set indicating that it is necessary to update this disk page. A disk store command is placed in the DQ.

The Strip Switcher Module (SS).

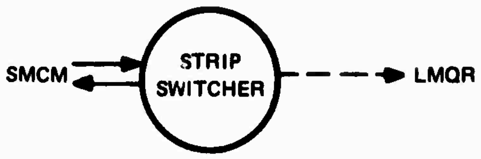

Figure 3.22 - SS Module

Called by the SMCM, this module places a strip switching command on the LMQ.

In a more sophisticated design, the SS could be required to clear the LMQ of all commands for the strip to be dismounted. This would involve freeing assigned buffers and disk pages and returning commands to the "DCTs.

The SWAP Module.

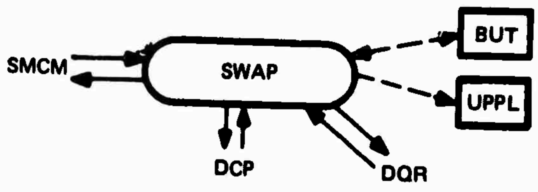

Figure 3.23 - SWAP Module

When called by the SMCM, the SWAP module will effect the freeing of all of the buffers being used by a particular user. By examining the BUT, the SWAP module can determine which buffers are assigned to the user (including those containing process pages), the CNT field is decremented and the user's UBB bit turned off. For those buffers with the change bit set, a write command to the disk page indicated in the BUTs dPPA field is placed in the DQ by the DQR. A RESTORE command for each non-process page is placed in the UDCT by the DCP. A SWAPIN command is also placed in the UDCT. The SMCM can restore the status existing before the SWAP by calling the DCX to execute these RESTORE and SWAPIN commands in the UDCT.

The Storage Manager Control Module (SMCM).

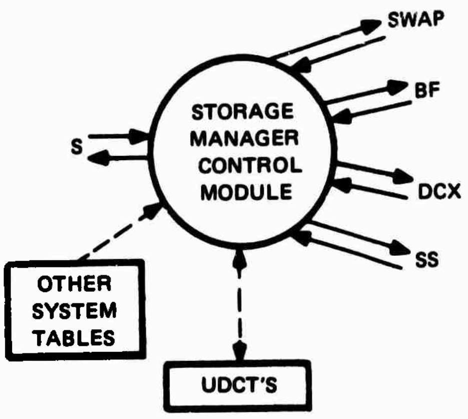

Figure 3.24 - SMCM

This module incorporates all of the decision making facilities of the storage manager. It is responsible for monitoring activity both in the UDCTs and in other parts of the system and initiating buffer allocation, swapping, and strip switching activity.

The SMCM may be looked upon as a "black box subroutine" capable of being implemented to the degree of sophistication deemed necessary. Several schemes have been considered for the SMCM. We will only discuss here the ways in which different kinds of information can be utilized by the control module.

By monitoring the UDCTs, the SMCM dictates the activities of the DCX in executing buffer requiring commands. Free buffers are always available since the SMCM controls the BF module. The SMCM is capable of recognizing a deadlock situation in buffer allocation and swapping one or more users out of core if necessary. Users requiring unmounted strips need not occupy core space and can be swapped in coordination with strip switching activity.

The SMCM may keep a number of internal tables (for example, a strip request table) which are referenced in making allocation decisions. For this reason, the DCP has been created so that it will be capable of updating these internal tables. The UDCTs themselves may be considered to be internal tables of the SMCM. This structure has not been more clearly defined as it will depend upon the particular strategies employed by the SMCM.

An efficient SMCM must monitor more than just the tables in the storage manager. Activity on the I/O queues, for example, may influence strip switching strategies since 10 seconds* worth of network output can be generated in a much smaller amount of time. Attention, too, should be paid to the process priorities kept by the supervisor. By means of these priorities, the supervisor is indirectly able to control storage manager scheduling of resources.

The SMCM will frequently be called by the supervisor to initiate storage manager activities.

---

* This is the approximate strip switching time for the laser memory.

The Disk and Laser Memory Operation Executors (DOX and LMOX).

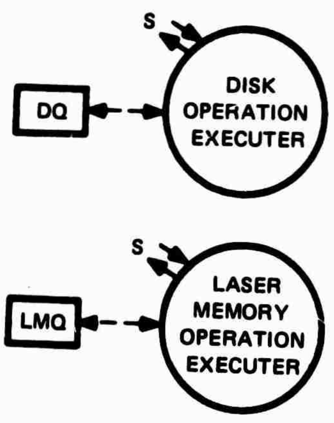

Figure 3.25 - DOX and LMOX Modules

These modules remove commands from the DQ and LMQ and perform the particular actions required. Upon completion of a command, they will set restart conditions for the completion routine. A portion of each of these modules will be interrupt driven. The LMOX will be implemented principally in the laser memory processors.

The Completion Routine (CR).

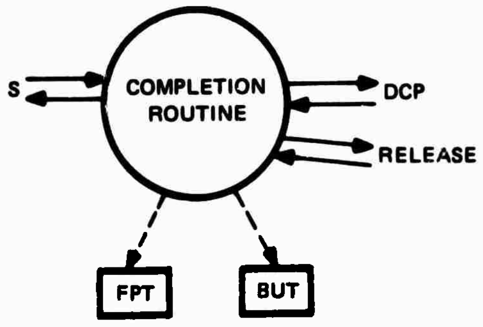

Figure 3.26 - CR Module

The completion routine performs clearing operations for completed commands from the DQ and LMQ.

Completed commands from the UDCTs are removed from the UDCTs. If the wait bit for the UDCT had been set and the last wait requiring command is removed from the UDCT, then the wait bit is cleared and the restart condition for the user is established.

Other operations are performed depending upon the particular command:

- disk stores to non-data pages: the change bit in the corresponding BUT entry is set to 0.
- stores to data pages on disk or laser memory: the disk page and buffer are RELEASEd.
- reads: the being read bit is set to 0.

Chapter 4
Input/Output Manager

4.1 Input/Output Manager Interface

Overview.
The I/O manager manages data transmission into and out of the datacomputer. The I/O manager consists of:

(1) device processes
(2) port monitors
(3) record conversion routines
(4) I/O subroutines
(5) port monitor creation routine.

Device Processes.
The device processes transmit physical records between the datacomputer and external sequential devices (e.g., the datacomputer's IMP, the Illiac, and standard magnetic tape). Input device processes transmit physical records from devices to physical record queues. Output device processes transmit physical records from physical record queues to devices. The device processes are independent processes, which are interrupt-driven.

Port Monitors.
A port monitor is created for each external sequential device used by a user process; the port monitor monitors data transmission between the user process and the external sequential device.

Each port monitor is an independent process. A port monitor buffers I/O between a user process and a device on record queues, which are maintained on pages in the basic page space, which are obtained by the port monitor from the storage manager.

An input port monitor issues commands to a device process to obtain physical records from a device, converts the records to logical records, and places the records on a queue, which is emptied by calls for input from the user process.

An output port monitor receives logical records from a user process, converts the records to physical records, places the records on a queue, and issues commands to a device process, which empties the queue, sending the records to a device.

A port monitor receives record conversion parameters from the user process.

I/O Subroutines.
I/O subroutines transmit data between user processes and port monitors. Calls to I/O subroutines are made by a user process in order to obtain input from a port monitor or in order to send output to a port monitor.

Record Conversion Routines.
Record conversion routines are called by the port monitors to convert logical records to physical records and vice versa.

Port Monitor Creation Routine.
A user process may call the port monitor creation routine in order to establish a connection between the user process and a device. (When an IMP input port monitor is created, the creation routine issues an ALLOCATE command to the IMP device process (Network Control Program).)

Interface Between Port Monitors and Storage Manager.
In order to manipulate scratch pages needed for its queues, a port monitor may issue ASSIGN, RELEASE, RESTORE, STANDBY, and WAIT commands to the storage manager (see Section 3.1). Each port monitor has a deferred command table, into which its RESTORE commands are placed by the storage manager.

Interface Between Port Monitors and Device Processes.
A port monitor may issue commands to a device process; commands are added to the device process' command queue.

A port monitor command to a device process has one of the following forms:

input port monitor:	READ <number records> <port> <pointer to list of in-core pages>*
output port monitor:	WRITE <number records> <port>

When executed, a command causes the indicated number of physical records to be transmitted through the port.

When a device process completes a port monitor command, the device process sets the port monitor's done-flag. If on input the records requested will overflow the pages provided or if on output the physical record queue does not contain the number of records specified, then the device process sets the port monitor's full/empty-flag.

Interface Between the Request Handler and Port Monitors.
A user process may issue to a port monitor the command

```
READ <n> RECORDS FROM <port>
```

or

```
WRITE <n> RECORDS TO <port>,
```

which cause logical records to be transmitted between the user process and the port monitor. For a READ, if the port monitor

* IMP input port monitors may issue the command "ALLOCATE <number messages> <port>", which allocates message blocks for messages to be received by the user process. The IMP input port monitor cannot issue a READ command because the data to be READ is not available from the IMP (and Network) on command from the datacomputer.

has at least <n> records, pointers to the <n> records are returned to the user process, and the records are removed from the port monitor's queue. Otherwise, <n> is posted in the port monitor's records-needed word and the supervisor is called. For a WRITE, <n> records are added to the port monitor's queue if the port monitor has space for <n> additional records. Otherwise, <n> is posted in the port monitor's records-needed word and the supervisor is called.

A user process may specify data descriptions and record conversion parameters to a port monitor.

A user process may create a port monitor, in order to connect the user process to a device, with a call to the port monitor creation routine:

```
     CREATE PORT MONITOR <user process id> <device> {INPUT/OUTPUT}.
```

Interface Between the Supervisor and the I/O Manager.
Device processes run on an interrupt basis determined by the supervisor. The supervisor schedules and runs the port monitors.

A PM may run when

```
       (1)  its records-needed word is not zero
or     (2)  its done-flag is set
or     (3)  its full/empty flag is set
or     (4)  a storage manager WAIT has completed.
```

A PM returns to the supervisor when

```
       (1)  its done-flag is not set
and    (2)  its full/empty-flag is not set
```

and (3) its records-needed word is zero

and (4) no records can be moved between LRQ and PRQ (the source queue is empty or the destination queue is full).

The I/O subroutines return to the supervisor and block a user process when the I/O subroutine needs additional records.

4.2 Input/Output Manager Architecture

Port and Port Monitor (PM).
A port is a connection between a user process and a device. The device may be the IMP, the Illiac, or standard magnetic tape. Each port has a port monitor.

A port monitor monitors data transmission for a port. Each PM is an independent process. Each PM has two queues: a physical record queue and a logical record queue.

Physical Record Queue (PRQ).
A PRQ is a list of indirect pointers to physical records. The indirect pointers to records are specified by record-address-blocks (RABs)*.

Record Address Block (RAB).
A RAB is a block of addresses. The entries in a RAB have the format:

    <type-bit> <address>

The last entry in a RAB has its type-bit set; in other entries the type-bit is cleared. The <address> is composed of two parts: page address and address-on-page (AOP).

* Not in the case of a PRQ for an IMP input port.

The AOP is an address relative to the beginning address of the core buffer into which the page specified by the page address has been RESTOREd. If the page address is given as zero, the AOP is relative to the beginning address of the core buffer in which the RAB is.

The first entry in a RAB is the address of the beginning of the record. The last entry of a RAB is the address of the end of the record. Other entries in a RAB are addresses of continuation pages of the record. Necessarily, the last two entries in a RAB have the same page address. The AOP of the last entry in the RAB must be greater than the AOP of the next-to-last entry.

Logical Record Queue (LRQ).
An LRQ is a list of pointers to logical records. An entry in the LRQ has one of the following forms:

00	<RAB-pointer>
01	<PPA>
10	<PPA>

The format-bits are 00 if the LRQ-entry is a RAB-pointer to a record; the format bits are 01 if the LRQ-entry points to a page that contains exactly one logical record of size 1K. The format-bits are 10 if the scratch page identified by <PPA> contains logical records in header format.

In header format, a page that contains logical records has a header. The header is a list of RABs, which identify the logical records on that page*. The header is written from the top of the page; logical records are added from the bottom of the page (see diagram).

* In some cases, a RAB may specify addresses not on the scratch page.

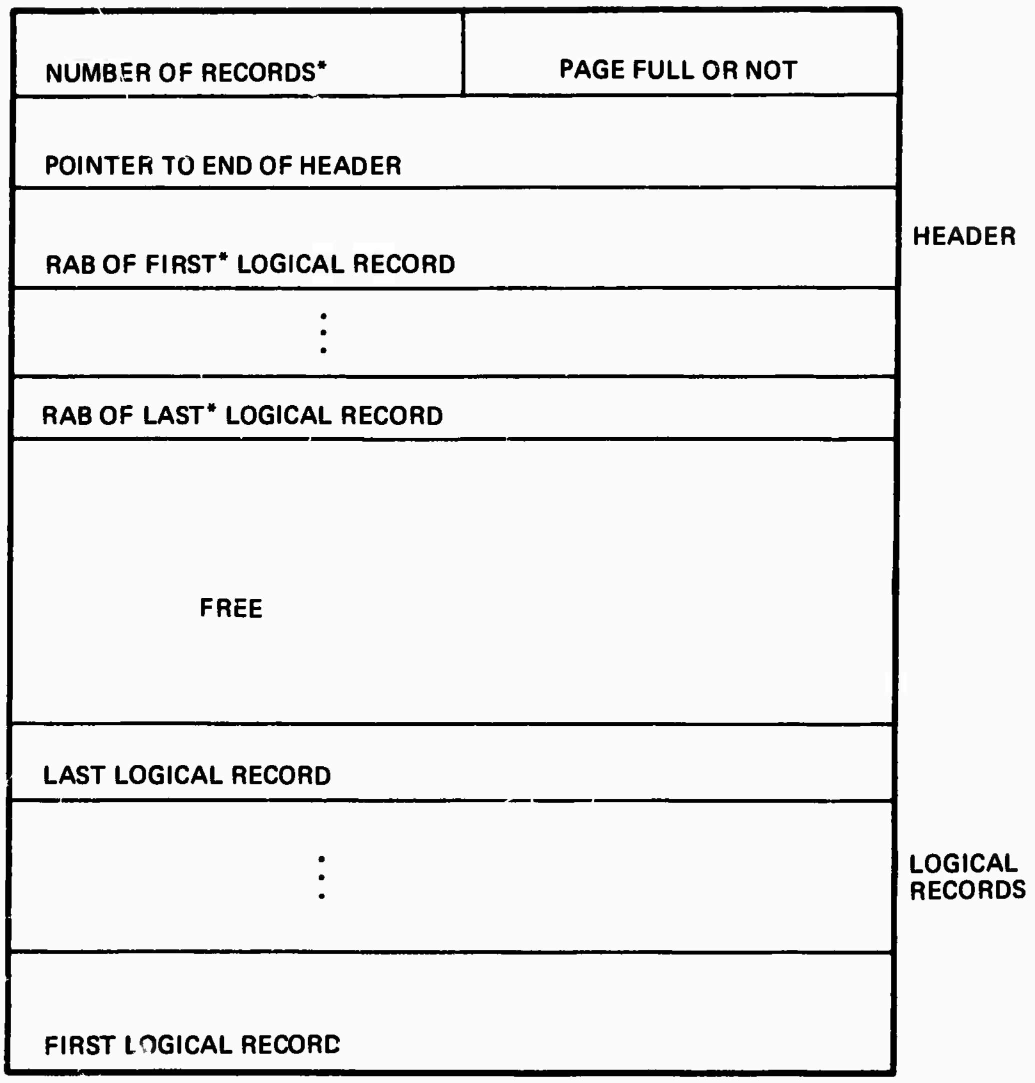

Figure 4.1 - Page Containing Logical Records

* on this page.

Port Monitor Actions.

A PM may:

1. Issue ASSIGN, RELEASE, RESTORE, STANDBY, and WAIT commands to the storage manager for pages for its queues (each PM has a deferred command table in the storage manager).
2. Issue commands to a device process to maintain its PRQ.
3. Transfer records from PRQ to LRQ (if input port) or from LRQ to PRQ (If output port).
4. Post information about queues.
5. Return to the supervisor.

Port Monitor Commands to the Storage Manager.

Commands to the storage manager from a PM are effected as they are for user processes (see Section 3.2).

Port Monitor Commands to Device Processes.

A command to a device process from a PM may have one of the following forms:

```
READ     <number records> <port> <pointer to list of
           in-core scratch pages>
WRITE    <number records> <port>
ALLOCATE <number messages> <IMP input port>*
RELEASE  <pointer to message block>*
```

Such commands when effected cause physical records to be added to or removed from the port's PRQ.

---

* For IMP input device process (Network Control Program) only.

Transfer of Records Between PRQ and LRQ.
The PM may call logical-record-to-physical-record or physical-record-to-logical-record conversion routines, which convert and transfer records between the PRQ and LRQ.

Port Queue Information.
A PM maintains for each of its queues:

the number of records in the queue,
a pointer to first record in the queue,
a pointer to last record in the queue,
the rate of filling the queue,
the rate of emptying the queue.

Device Processes.
Device processes read physical records from or write records to a device. Each device process has a command queue, into which port monitors put their commands.

Each PM has a done-flag and a full-empty-flag, which may be set by the device process. The done-flag is set when the device process completes the PM's command. The full/empty-flag is set when a READ command has been issued by a PM and the scratch pages for physical records become full or when a WRITE command has been issued by a PM and its PRQ becomes empty.

I/O Subroutines.
I/O subroutines transmit logical records between a user process and the LRQs of ports connecting the user to devices.

Records may be transmitted between a user process and LRQs in two forms: copy-form and no-copy-form. In both forms, the receiver of the data receives a pointer (a RAB) to the record. In copy-form, the pointer points to a copy of the record; in no-copy-form, the pointer points to the original record. (The no-copy-form requires that both the user process and PM process share a scratch page.)

I/O Subroutines for Input.
An I/O subroutine for input is invoked by the request handler when a "READ <n> RECORDS FROM <port>" is encountered in the user process. If the LRQ contains at least <n> records, the subroutine returns pointers for <n> records from the LRQ of the port <port> and removes those records from the LRQ. Otherwise, the subroutine posts the number <n> in the PMs records-needed word and gives a WAIT command to the supervisor. The user process is blocked until the LRQ contains <n> records in core.

I/O Subroutines for Output.
An I/O subroutine for output is invoked by the request handler when a "WRITE <n> RECORDS TO <port>" is encountered in the user process. The subroutine adds <n> records to the LRQ if the LRQ can hold <n> more records. Otherwise, the subroutine posts the number <n> in the PMs records-needed word and gives a WAIT to the supervisor. The user process is blocked until the LRQ can hold <n> more records.

Special Considerations for the Illiac Port.
The Illiac Port Monitors issue special ASSIGN and RESTORE commands which cause scratch pages on the fast disk to be ASSIGNed and which RESTORE those scratch pages in special datacomputer/ Illiac shared core.

In Illiac dump mode, the Illiac port monitors maintain only an abbreviated PRQ and no LRQ. The abbreviated PRQ contains the disk address of the first page of the dump and the number of pages dumped.

Special Considerations for IMP Input Ports.
The Network Control Program (NCP) is the datacomputer's device process for the IMP. For buffering input from the network, the NCP has a Message Queue (MSQ). The MSQ is a set of pages divided into blocks of about 8000 bits. Associated with the MSQ is a bit map of the MSQ blocks. The NCP also has an allocation table (AT), that indicates how many blocks a user has been allocated and how many blocks a user has in use.

A PM for IMP input is unlike other input PMs. An IMP input PM has a special PRQ and does not manage scratch pages for its physical records. The PRQ for an IMP input PM is not a list of RABs but rather a list of pointers to blocks (messages) in the MSQ. As a message for a port is received, the NCP adds to the port's PRQ a pointer to the block containing the message; the block's use-bit is set in the MQ bit-map.

An IMP input PM may issue two commands to the NCP (IMP device process): allocate blocks and release blocks.

The "allocate" command is used to enter in the AT the number of blocks allocated to a port. The allocation of blocks to a port may require that the NCP call the MSQ page allocator module to obtain additional core space or to release unneeded core space for the MSQ. Block allocation is similar to ASSIGN-RESTORE for non-IMP-ports.

The "release block" command is used to clear in the MSQ-map the use-bit, of a block which has been processed by a physical-to-logical conversion routine. Block-release is similar to RELEASE for non-IMP-ports.

Chapter 5
Supervisor

5.1 Supervisor Interface

In the datacomputer the supervisor, storage manager, and I/O manager together replace the generalized time sharing operating system. The supervisor has modules for the following jobs:

- management of clock, priority interrupt system and protection and relocation hardware
- operator communication
- program loading
- performance monitoring and accounting
- process scheduling.

Scheduled processes run when granted CPU time by the supervisor. User processes, the sequential I/O port monitor processes, and the laser memory strip scheduling process are examples of scheduled processes.

Interrupt processes run when started by the interrupt hardware. They control I/O devices and handle unusual conditions. An interrupt process normally has the CPU for a very short time, performs some real-time function, may post some information for scheduled processes, and then restarts the process it interrupted or passes control to the supervisor.

In the following, the five commands that can be issued to the supervisor are each followed by a description of the supervisor action taken for the command.

1. WAIT <restart conditions>

The supervisor may start up another process if desirable; the currently running process must wait until the specified restart conditions are satisfied. The supervisor grants the CPU to another process if any can be run and if process switching has not been disabled.

Allowable restart conditions are:

a. any of a set of flags must be on
b. all of a set of flags must be on.

2. POST <pointer to flag>

The supervisor sets the indicated flag and starts up another process if it is desirable. This command is commonly issued by routines surrendering a scarce resource or signaling the completion of an important event.

3. INTERRUPT <machine state information, process id, entry point>

The supervisor saves the state of the most recently running scheduled process and starts up the process named at the entry point specified, after all waiting interrupts have been served. After an interrupt process has seized the CPU, it may issue an INTERRUPT command to start a process.

4. INITIATE <process state table entry>

The supervisor creates a process with the initial state specified in the entry. The supervisor adds the entry to the process state table and puts the process on the process schedule. If the process' restart conditions are satisfied, it can be started immediately.

5. TERMINATE <process id>

The supervisor deletes the indicated process from the process state table and releases all of its resources.

5.2 Supervisor Architecture

Hardware Management.
Clock management routines provide time of day and interval timing services for scheduling, accounting, and performance monitoring routines.

The priority interrupt system management routines control the priority channels and their service locations and thus control the I/O device processes.

Operator Communication.
The computer operator must have the ability to control the datacomputer while it is running production and to perform a wide variety of functions while the datacomputer is being modified or running diagnostics.

There is a simple language for the operator to use and a module to process it. Commands and queries are processed at a time convenient for the supervisor, probably immediately before process selection. Diagnostics and status information are also output by this module.

Program Loading.
The supervisor has two special requirements for program loading: loading itself and reloading the datacomputer programs carefully in case of catastrophe. For loading itself, the supervisor can use the standard methods employed by other PDP-10 monitors. Recovering in case of catastrophe requires special techniques to salvage as much of the file maintenance activity as possible and to maintain connection to the current users.

Program loading is done by a special supervisor module because the storage manager (which normally does disk I/O) and the sequential I/O manager are among the programs to be loaded.

Fetching of less frequently used program modules from disk during datacomputer operation is handled by the storage manager, since the programs are in a standard disk file.

Performance Monitoring and Accounting.
This module gathers necessary data for analysis of the design, tuning of algorithms, system evaluation, and file reorganization and disposition.

Process Scheduling.
The supervisor establishes the schedule by which processes are to be run. The process schedule may be modified on any call to the supervisor. After a call to the supervisor (i.e., after a WAIT was issued to the storage manager, an I/O subroutine needed records, or a quantum of time elapsed), the process to be run next is determined by the process schedule.

The process scheduling module establishes the schedule by consulting the process' state tables and static and dynamic priority information.

A process state table indicates, among other things, whether a process may be run and upon what other processes the process is waiting.

Static priority information reflects unchanging priority factors such as:

1. Whether a user process corresponds to a local or remote user.

2. Whether a user process has special priority inherently (as for Illiac fast dump) or imposed.

Dynamic priority information reflects changing priority factors and the relationships among processes. For example, the high priority of a user process, which is waiting for input from a port monitor may influence the priority of the port monitor.

Appendix
Glossary of Acronyms

AOP	address on page
AT	allocation table
BF	buffer freer module
BUT	buffer usage table
CNT	count
CPU	central processing unit
CR	completion routine
dPPA	physical page address (on disk)
DCP	deferred command poster
DCX	deferred command executor
DOX	disk operation executor
DQ	disk queue
DQR	disk queuer module
FAM	file allocation module
FDH	file directory handler
FPT	free page table
FR	future read
FRT	future read table
ICM	in-core map
IMP	interface message processor
IOM	imput/output manager
LMOX	laser memory operation executor
LMQ	laser memory queue
LMQR	laser memory queuer module
LPA	logical page address
LPAM	logical-physical address mapper
LPM	logical-physical map
LRQ	logical record queue
MCD	mapped command dispatcher
MSQ	message queue
MQ	mapper queue

NCP	network control program
PAW	page address word
PID	page identifier
PM	port monitor
PPA	physical page address
PRQ	physical record queue
RAB	record address block
RH	request handler
RQ	read queue
S	supervisor
SM	storage manager
SMCI	storage manager control interface
SMCM	storage manager control module
SNM	strip-not-mounted
SS	strip switcher module
SPT	shared page table
UBB	usage bit block
UDCT	user's deferred command table
UPPL	user's process page list
WRCRT	wait-requiring command results table
XPAW	extended page address word
XPDA	extend process data area